

FIG. 1

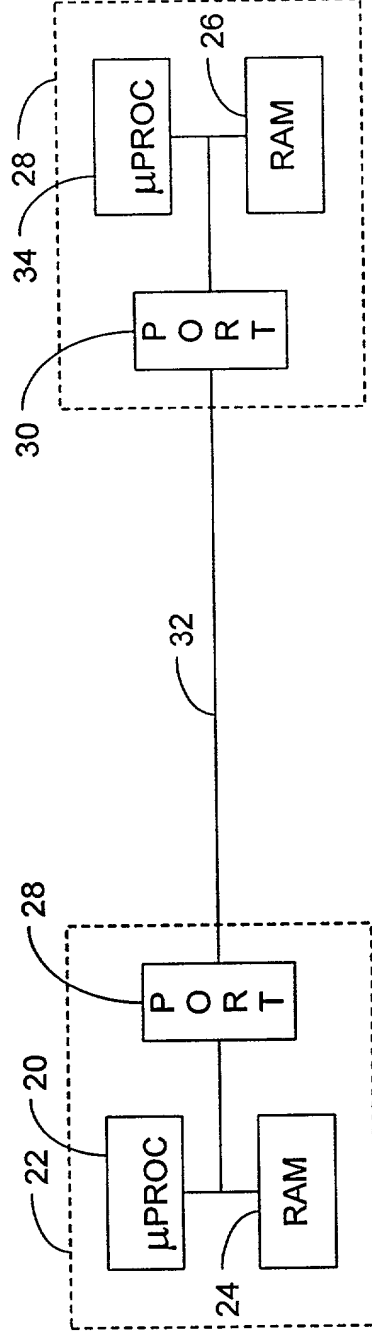


FIG. 2

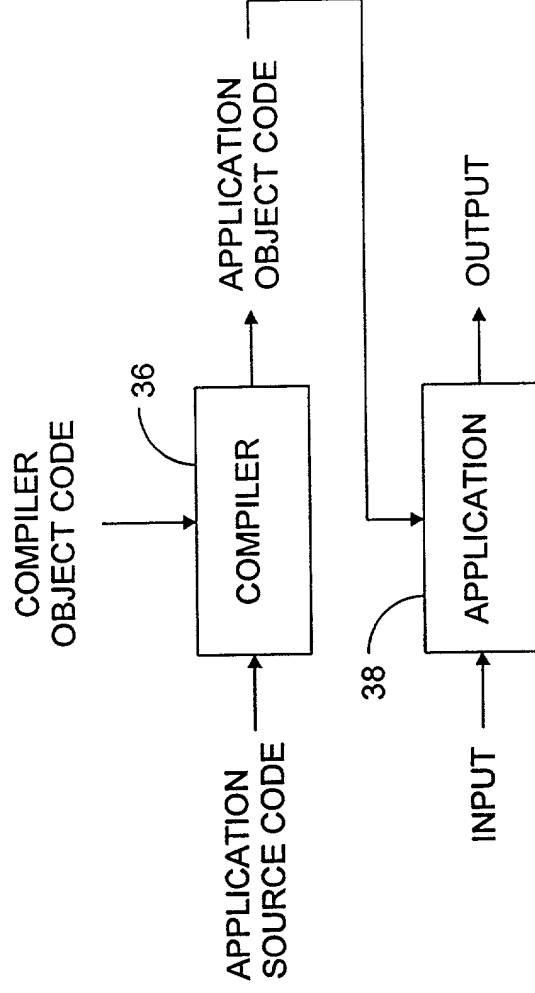


FIG. 3

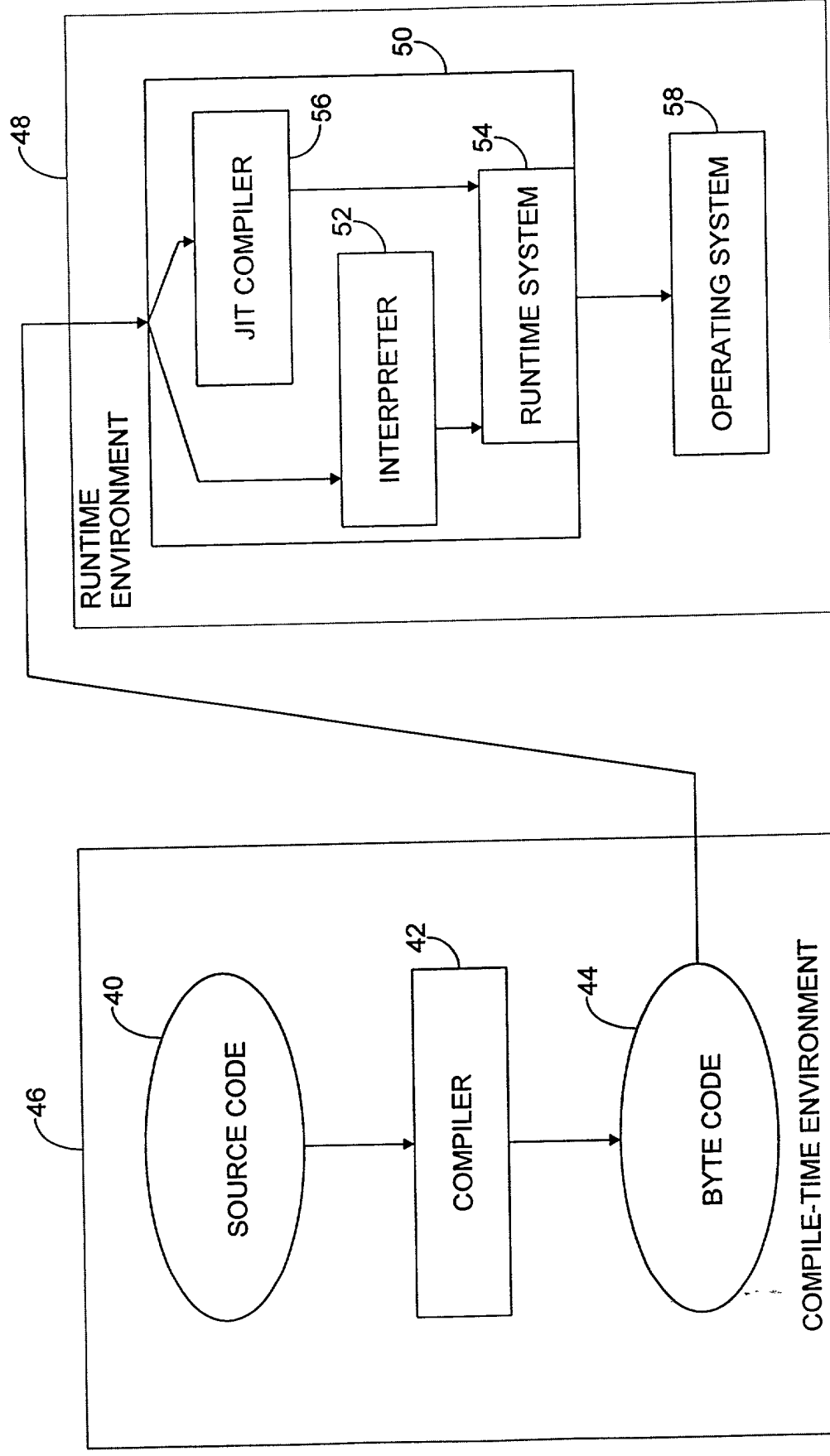


FIG. 4

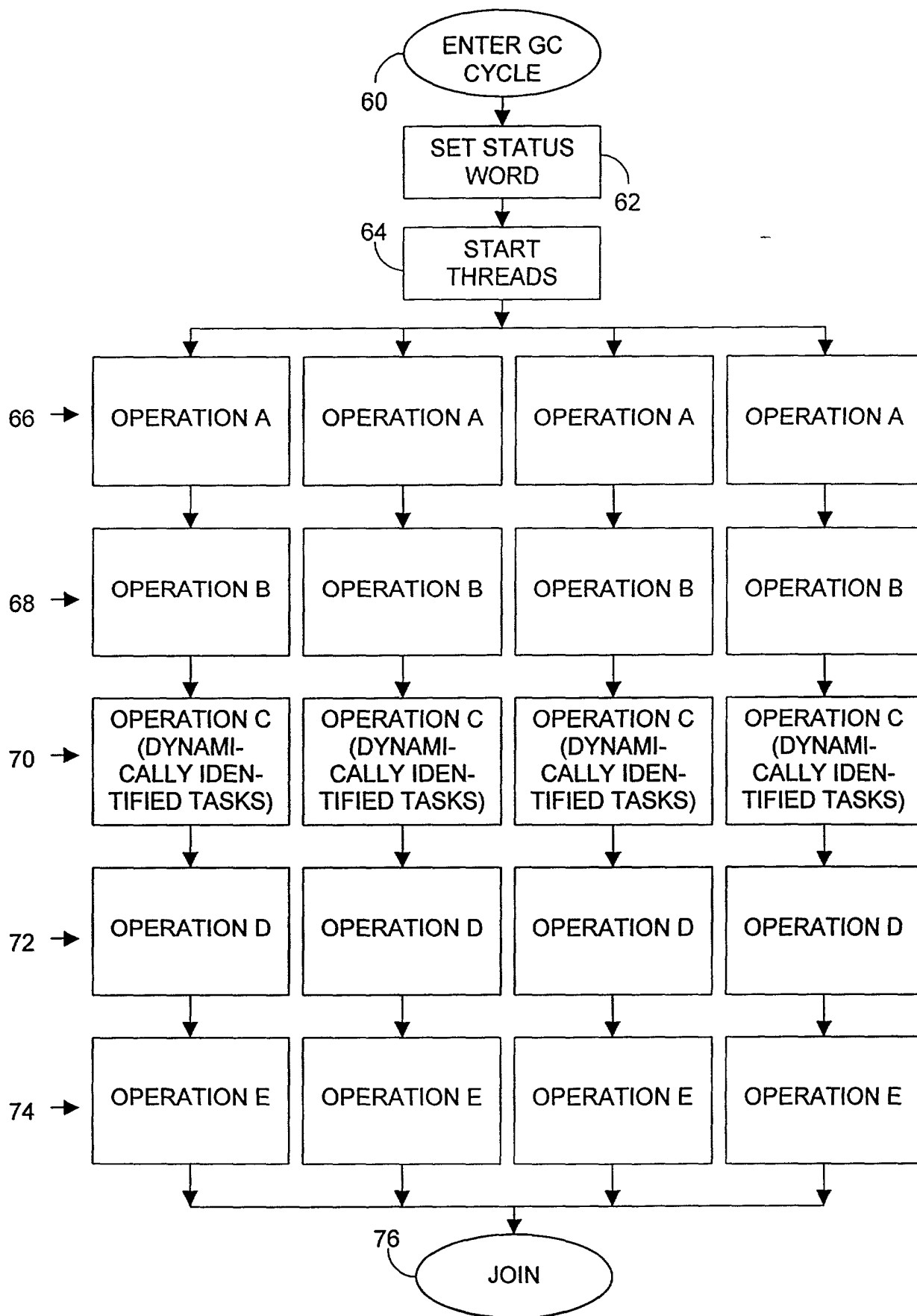


FIG. 5

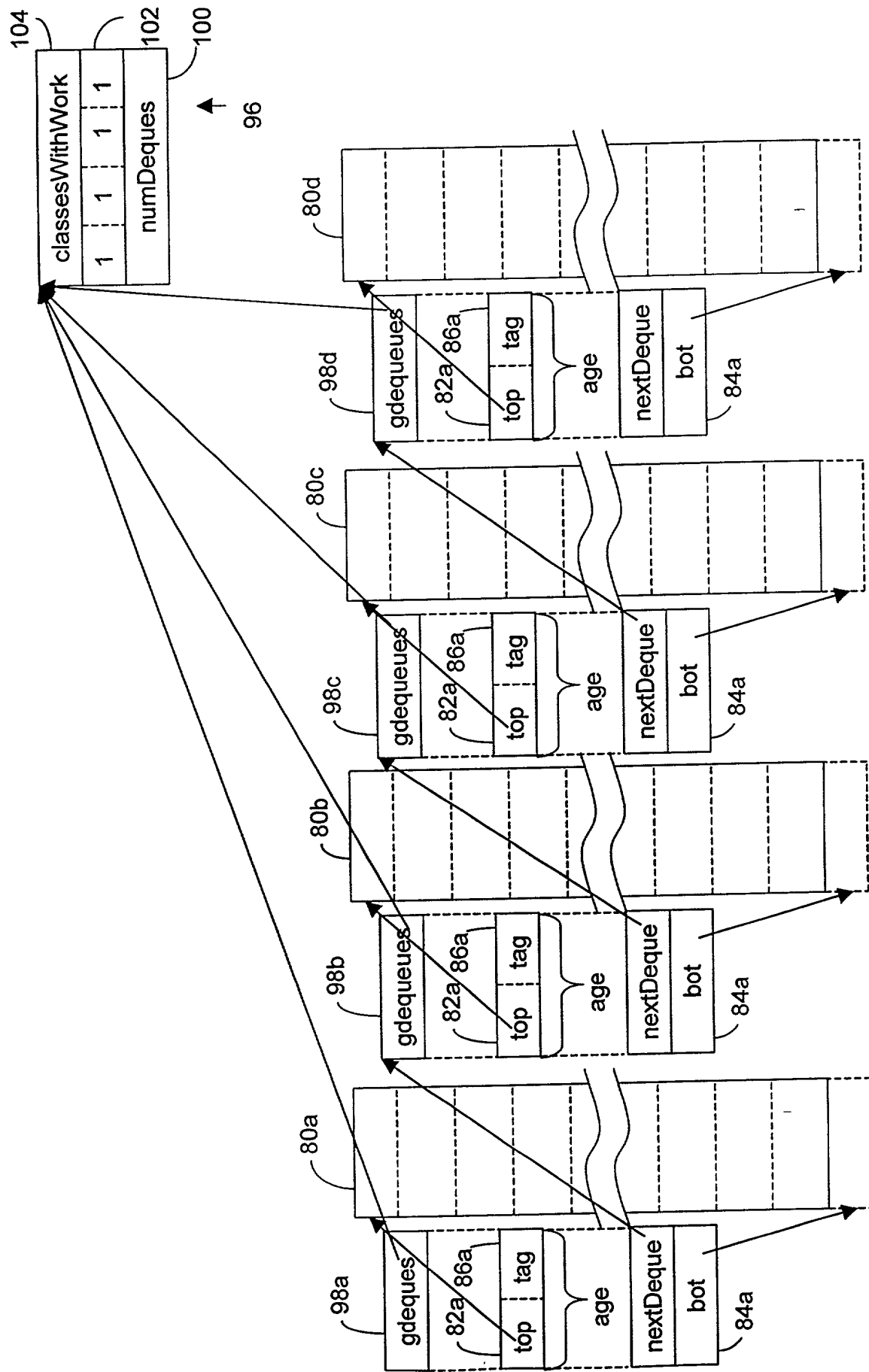


FIG. 6

```

1  java_lang_Object* popTop ()
2  {
3      oldAge = age;
4      localBot = bot;
5      if (localBot <= oldAge.top)
6          return NULL;
7      task = deq[oldAge.top];
8      newAge = oldAge;
9      newAge.top++;
10     cas(age, oldAge, newAge); /*atomic compare-and-swap*/
11     if(oldAge == newAge)
12         return task;
13     return NULL;
14 }

```

FIG. 7

```

1  void pushBottom(java_lang_Object* task)
2  {
3      localBot = bot;
4      deq[localBot] = task;
5      localBot++;
6      bot = localBot;
7  }

```

FIG. 8

```

1  java_lang_Object* popBottom()
2  {
3      localBot = bot;
4      if (localBot == 0)
5          return NULL;
6      localBot--;
7      bot = localBot;
8      task = deq[localBot];
9      oldAge = age;
10     if (localBot > oldAge.top)
11         return task;
12     bot = 0;
13     newAge.top = 0;
14     newAge.tag = oldAge.tag + 1;
15     if (localBot == oldAge.top) {
16         cas(age, oldAge, newAge)
17         if (oldAge == newAge)
18             return task;
19     }
20     age = newAge;
21     return NULL;
22 }

```

FIG. 9

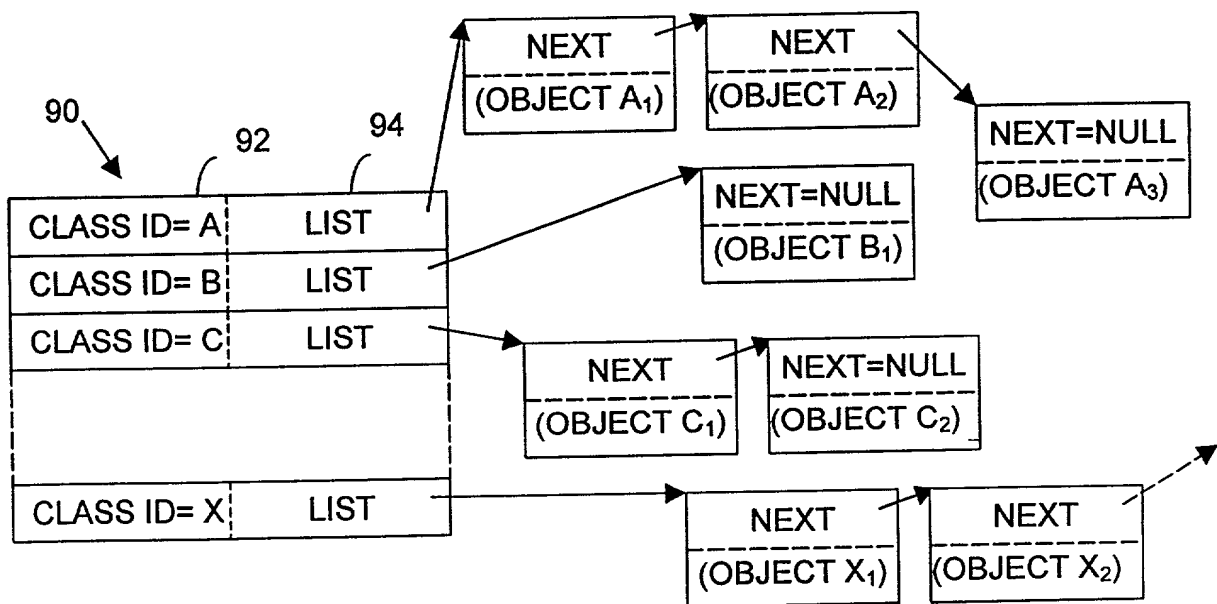


FIG. 10

```

1  java_lang_Object *dequeFindWork(localDeque *dq) {
2      java_lang_Object *result = findWorkHelper(dq);
3      globalDeques *gdqs = dq->gdeques;
4      if (result == NULL) {
5          mark_self_inactive(dq->index, &gdqs->statusBitmap); /* You have no work */
6      }
7      while (result == NULL) {
8          if (!gdqs->statusBitmap) return NULL; /* No one has any work. Terminate. */
9          poll(NULL, NULL, 0);
10         if (checkForWork(dq)) { /* You don't have any work, but there is some either
11             on the overflow queue, or in another thread's work
12             queue */
13             mark_self_active(dq->index, &gdqs->statusBitmap); /* Looking for work */
14             result = findWorkHelper(dq);
15             if (result == NULL) {
16                 mark_self_inactive(dq->index, &gdqs->statusBitmap);
17             }
18         }
19     }
20     return result;
21 }

1  Java_lang_Object *findWorkHelper(localDeque *dq) {
2      java_lang_Object *task = findWorkInOverflowList(dq);
3      if (task == NULL) {
4          task = stealWork(dq);
5      }
6      return task;
7  }

```

FIG. 11


```

1 static java_lang_Object *stealWork(localDeque *dq) {
2     globalDeques *gdqs = dq->gdeques;
3     int degree = gdqs->numDeques;
4     int iterations = 2 * degree;
5     int i = 0;
6     while (i++ < iterations) {
7         localDeque *dqToSteal = pickQueueToStealFrom(gdqs, dq);
8         if (dqToSteal->bot > dqToSteal->age.top) {
9             java_lang_Object *task = (java_lang_Object *)popTop(dqToSteal);
10            if (!task) poll(NULL, NULL, 0);
11            else return task;
12        }
13    }
14    return NULL;
15 }

```

FIG. 12

```

1 static bool_t checkForWork(localDeque *dq) {
2     globalDeques *gdqs = dq->gdeques;
3     return gdqs->classesWithWork || peekDeque(dq);
4 }

1 static bool_t peekDeque(localDeque *dq) {
2     globalDeques *gdqs = dq->gdeques;
3     int degree = gdqs->numDeques;
4     int i;
5     for (i = 0; i < 2 * degree; i++) {
6         localDeque *dqToPeek = pickQueueToStealFrom(gdqs, dq);
7         if (dqToPeek->bot > dqToPeek->age.top) {
8             return TRUE;
9         }
10    }
11    return FALSE;
12 }

```

FIG. 13